AD-A085 664 VIRGINIA POLYTECHNIC INST AND STATE UNIV WASHINGTON --ETC F/G 9/2
CMS RATFOR USER'S MANUAL.(U)
JUL 79 S M CHOQUETTE, R J ORGASS AFOSR-79-0021
UNCLASSIFIED VPI/SU-TM-79-5 AFOSR-TR-80-0444 NL

1 of 1
AD A
085664

END
DATE
FILMED
7-80
DTIC

(18) AFOSR TR- 80-0444 (19) (12) EXTENSION DIVISION

VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY

LEVEL II

DEPARTMENT OF COMPUTER SCIENCE
GRADUATE PROGRAM IN NORTHERN VIRGINIA

P. O. Box 17186
Washington, D. C. 20041
(703) 471-4600

CMS RATFOR USER'S MANUAL†

(15) AFOSR-79-0021

(10) Stephen M. Choquette
~~and~~
Richard J. Orgass

(9) Technical ~~Memorandum~~ No. 79-5

July 1, 1979

(11) 1 Jul 79

DTIC ELECTE JUN 1 6 1980 D C

ADA085664

ABSTRACT

RATFOR is a preprocessor for Fortran that provides modern control structures and a substantial improvement in the syntax of Fortran programs. The output of RATFOR is a Fortran program that is compiled by the Fortran processors and then executed.

The RATFOR preprocessor provides statement grouping, IF-ELSE structures and four loops: DO, FOR, WHILE and REPEAT-UNTIL.

RATFOR source text is free format with multiple statements on a line. Upper and lower case letters are treated as upper case letters except in character constants. There is an include facility so that large programs can be constructed out of a multitude of small files without using the system editor. RATFOR accepts files consisting of fixed length 80 column records with imbedded tabs.

(14) VPI/SU-TM-79-5

The program described here is an adaptation of the original RATFOR processor written at Bell Laboratories for use in the CMS environment.

This manual is a complete set of instructions for using the preprocessor. It includes a description of the RATFOR syntax, a detailed explanation of the preprocessor error messages and directions for using the processor. A detailed description of the preprocessor, which is not needed for general users, is Technical Memorandum No. 79-4 and may be obtained from the second author at the above address.

---

* Research sponsored by the Air Force Office of Scientific Research, Air Force Systems Command, under Grant No. AFOSR-79-0021. The United States Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation hereon.

† The information in this document is subject to change without notice. The authors, Virginia Polytechnic Institute and State University, the Commonwealth of Virginia and the United States Government assume no responsibility for errors that may be present in this document or in the program described here.

80 6 11 022

Approved for public release; distribution unlimited.

DDC FILE COPY

411 731

Copyright, 1979

by

Stephen M. Choquette

and

Richard J. Orgass

General permission to republish, but not for profit, all or part of this report is granted, provided that the copyright notice is given and that reference is made to the publication (Technical Memorandum No. 79-5, Department of Computer Science, Graduate Program in Northern Virginia, Virginia Polytechnic Institute and State University), to its date of issue and to the fact that reprinting privileges were granted by the authors.

AIR FO[illegible] (AFSC)
NOTI[illegible]
This [illegible] and is
appr[illegible] -12 (7b).
Distr[illegible].
A. D. [illegible]
Technical [illegible] Officer

RATFOR User's Manual

* * * TABLE OF CONTENTS * * *

Summary of RATFOR Features

With the growing concern for the cost of program development, the computer industry has seen a shift towards programming languages that emphasize an organized approach to program development. RATFOR is a new, rational approach to programming in FORTRAN; the language offers RATFOR users the universality and efficiency of FORTRAN, while providing decent program flow control structures. RATFOR, implemented as a preprocessor to FORTRAN, offers the busy programmer statement grouping, IF-ELSE segmenting, and DO, FOR, WHILE, and REPEAT-UNTIL loops. Additional RATFOR features and keywords make code maintenance a less painful task.

This paper is the User's Manual for the RATFOR preprocessor on the IBM CMS timesharing system. Included in this paper is a language description of RATFOR, a discussion of how to use RATFOR on CMS, sample terminal sessions and a detailed explanation of the RATFOR error messages. Within the language description section is a quick reference guide listing the keyword syntax and RATFOR cosmetic features.

Accession For
NTIS GRA&I [✓]
DDC TAB ☐
Unannounced ☐
Justification______

By______
Distribution/______
Availability Codes

Dist	Avail and/or special
A	

Description of RATFOR

RATFOR is an attempt to hide the worst of FORTRANs defficiencies (primarily hard to understand code), while retaining the advantages of the language (universality, portability, efficiency). RATFOR offers the user powerful program flow control statements without the FORTRAN necessities of GOTOs and labeled statements. Additionally, the RATFOR language provides the user with many "pretty print" features so that a program in RATFOR would be easier to understand and maintain.

The remainder of the Language Description section will explain both the syntactic and cosmetic features of RATFOR. The program flow control structures are typical of the newer high level programming languages. For a quick reference to RATFOR, flip to the Quick Reference Guide at the end of the Language Description section.

Statement Grouping

Often a programmer will want to group a sequence of statements together for execution on a certain condition. Generally, the programmer needs something to say "if some condition, do these things." In RATFOR, statements can be grouped by enclosing them within brackets. For example,

```
IF (A >= 1ØØ)
   { C = C + 1
     SUM = SUM + A
     L = Ø
    }
```

The example above is a legal RATFOR program segment. Note that the brackets denote a sequence of statements to execute when A >= 1ØØ. The brackets perform the same function as the PL/1 DO/END sequence or the PASCAL BEGIN/END sequence.

To the avid FORTRAN programmer, a few things will stand out in the above example. First, RATFOR allows free form input; statements can occur in any column. When RATFOR encounters a statement starting with an all-numeric field, the processor assumes the field is a FORTRAN label and places it in columns 1-5 of the output. Next, the FORTRAN user will observe that '>=' is not a legal FORTRAN boolean relational operator. RATFOR will translate the more understandable relational operators (>,>=,==,<,<=,~,~=,&,|) into their FORTRAN equivalent. The last observation the FORTRAN user will make is the semicolon. Being a free form language, RATFOR allows more than one statement per line. The additional statements must be separated by a semicolon. When only one statement is on a line, the semicolon is optional. Thus, the previous example could be written as

```
        IF(A>=100) {C=C+1;SUM=SUM+A;L=0;}
```

Obviously the first form is easier to understand. One final comment is needed; when a statement is obviously not finished on one line, RATFOR assumes it will be continued on the next line. No character is needed in column 6.

ELSE Clauses

Occassionally the FORTRAN programmer will want to say "if some condition, do these things, otherwise do these." The ELSE clause provides this option. Naturally the ELSE clause may be left off. The full format of the IF statement is

```
        IF (legal FORTRAN condition)
           RATFOR statement
         ELSE
           RATFOR statement
```

The RATFOR statement can be in one of three forms: 1) a single FORTRAN statement (no brackets needed), 2) a bracketed segment of RATFOR and FORTRAN statements, or 3) another RATFOR keyword.

Notice that the third case allows us the option of nested IFs. A legal sequence of RATFOR statements is

```
        IF (A == B)
           CTR1=CTR1+1
         ELSE IF (A>B)
              CTR2=CTR2+1
        ELSE
              CTR3=CTR3+1
```

Like many languages allowing the IF-ELSE construct, the question arises of which IF does the ELSE match with. This ambiguity is resolved by matching the ELSE with the last unmatched IF. Because RATFOR allows free form input, the user should use indentation to clarify the program listing.

The DO Statement

The RATFOR DO statement is very similar to the FORTRAN DO statement.

The DO Statement

The major omission is the lack of a statement number. The format of the RATFOR DO statement is

```
          DO legal-FORTRAN-DO-test
             RATFOR statement
```

As before, the RATFOR statement can be a single statement or a sequence of RATFOR statements in brackets. Since the RATFOR statement can be another RATFOR statement, the following sequence is legal.

```
          IF (A ~= B)
             DO I=1,5
                IF (SWITCH(i) == 1)
                   CTR1=CTR1+1
                 ELSE
                   CTR2=CTR2+1
```

Notice that in each case, a single RATFOR statement follows the IF and DO statements. Occassionally brackets will help clarify the listing, although they are not necessary.

<u>The BREAK Keyword</u>

The BREAK keyword provides a way of exiting a loop without using the FORTRAN GOTO statement. BREAK can be followed by an integer (BREAK N) specifying how many levels of looping to exit from. The following sequence locates the first non-blank character in a string array

```
          DO I=1,8Ø
             IF (STRING(I) == BLANK)
                BREAK
```

BREAK jumps to the statement after the end of the specified loop.

'<u>The NEXT Keyword</u>'

Like BREAK, the NEXT keyword provides a means of loop control without the GOTO statement. NEXT jumps to the iteration step of the specified loop. NEXT can also be followed by an integer (NEXT N) giving the loop level to go to

```
          DO I=1,8Ø
             { IF (STR(I)==BLANK)
                  NEXT
               STR(I)=STAR
              }
```

This sequence of code sets all non-blank characters in a string array to '*'. STAR is assumed defined above.

The WHILE Statement

WHILE provides a more powerful looping structure than the simple DO loop. The syntax of the WHILE statement is

```
WHILE (legal FORTRAN condition)
    RATFOR statement
```

Notice that the WHILE loop checks the FORTRAN condition at the start of the loop so that the loop may be executed zero times. The sequence

```
I=8Ø
WHILE(STR(I)~=BLANK)
   I=I+1
```

locates the last non-blank character in an input card of 8Ø characters. Of course the NEXT and BREAK statements can occur within a WHILE loop. The NEXT statement in a WHILE loop goes to the test condition.

The FOR Statement

The FOR statement is yet another powerful RATFOR loop structure. The syntax of the FOR statement is

```
FOR (initial; condition; increment)
    RATFOR statement
```

where initial is any one FORTRAN statement, condition is the stopping condition and increment is the final step (a single statement) in the loop. Any of the fields can be null so long as the semicolon delimiter is present. Our last non-blank character example above is

```
FOR (I=8Ø; STR(I)~=BLANK; I=I-1)
    ;
```

Notice that the actual loop portion is the null statement. This is because everything we will want to do is in the FOR statement. As may be obvious by now, certain loop constructs work better in different situations. Choose the best and simplest for your work. The NEXT statement in a FOR loop goes to the increment step of the loop.

The REPEAT-UNTIL Statements

The REPEAT-UNTIL construct is the last loop control structure in RATFOR. It provides a means of checking the exit condition at the bottom of the loop. Recall that WHILE and FOR check at the top of loops. The syntax of the statement is

```
REPEAT
   RATFOR statement
 UNTIL (legal FORTRAN condition)
```

The UNTIL is optional and, if omitted, provides an infinite loop. Of course the programmer will want to get out of the loop using BREAK, STOP, or RETURN. Caution should be used with the REPEAT-UNTIL construct as it does not test for the null case.

The RETURN Statement

The standard FORTRAN RETURN mechanism uses the function name to return a value. This is allowable in RATFOR as well as expressions of the form

```
RETURN (value)
```

If there are no parentheses, a normal RETURN is made.

The DEFINE Statement

The DEFINE statement is not an executed RATFOR statement; no FORTRAN code is generated. The statement allows the user to create program definitions to make his program more understandable. The syntax is

```
DEFINE (name, definition)
              or
DEFINE  name, definition
```

Every occurrence of name in the user source code is immediately replaced by the definition. Optimally, DEFINEs should be at the start of the source code to clarify their use. The name portion can be arbitrarily long and must start with a letter. The DEFINE statement can be used to define global constants as follows

```
DEFINE (YES,1)
DEFINE (NO,Ø)
```

With the above statements, we can now say

```
IF (ISMTQ == 1)
   RETURN (YES)
 ELSE
   RETURN (NO)
```

The INCLUDE Statement

The INCLUDE statement inserts files directly in the RATFOR source code input. The statement

```
        INCLUDE RATCMN
```

inserts the CMS file RATCMN (possibly containing COMMON blocks) into the user source code in place of the INCLUDE statement. Thus, the programmer would type the program COMMON blocks once into the RATCMN file and then, in each subroutine needing the COMMON blocks, insert "INCLUDE RATCMN." Of course changes to the RATCMN file would affect every subroutine having the INCLUDE statement. The syntax of the statement is

```
        INCLUDE  FN  FT  FM
```

Where FN is the CMS file name (8 characters maximum), FT the CMS file type (optional - Default RATFOR), and FM the CMS file mode (optional - Default A1).

RATFOR Cosmetic Features

As mentioned before, RATFOR provides many cosmetic features to allow the user a sharp looking listing. In addition to a sharper listing, the use of cosmetics makes the source listing more readable and easier to maintain.

First, RATFOR allows free format input. Statements can occur anywhere on a line. If more than one statement is on a line, they must be separated by a semicolon. Blank lines are ignored. The user need not worry about a long statement continuing to a new card; RATFOR can make a fair estimate whether the statement is a continuation. Lines ending with any of the characters

```
        = + - * , | & (
```

are assumed to be continued on the next line.

The next cosmetic feature is RATFOR commenting. Comments start with a # and can occur anywhere in a line. Thus, comments can occur next to source statements. Comments are assumed to continue until the end of the card.

RATFOR will perform translation services for the user whenever they are needed, excepting within single or double quotes:

==	to	.eq.	~=	to	.ne.
>	to	.gt.	>=	to	.ge.
<	to	.lt.	<=	to	.le.
&	to	.and.	|	to	.or.
!	to	.not.	~	to	.not.

Additionally, the statement grouping brackets can be either { and }, [and], or $(and $).

One important cosmetic feature is that RATFOR input can be in upper and lower case. Anything not within single or double quotes is translated to upper case for the CMS FORTRAN compiler.

Lastly, text within matching single or double quotes is converted to its Hollerith equivalent ('string'=6Hstring). Within quoted strings, the backslash '\' serves as an escape character; the next character is taken literally. This way a single quote can be entered as "\'" .

Quick Reference Guide to RATFOR

(Keyword Syntax)

BREAK keyword

```
    BREAK N             (N=1 by default)
```

Exits from N levels of enclosing loops.

DEFINE statement

```
    DEFINE (defined name, defined value)
                        or
    DEFINE defined name, defined value
```

Defined name may be arbitrarily long & must start with a letter.

DO statement

```
    DO  legal-FORTRAN-DO-test
        RATFOR statement
```

FOR statement

```
    FOR (initial; condition; increment)
        RATFOR statement
```

```
    Initial - any single FORTRAN statement
    Condition - any legal RATFOR condition
    Increment - any single FORTRAN statement
```

IF statement

```
    IF (legal FORTRAN condition)
       RATFOR statement
     ELSE
       RATFOR statement
```

The ELSE is optional and is matched with the last IF.

INCLUDE statement

```
INCLUDE    FN     FT     FM

FN - CMS file name (8 characters maximum)
FT - CMS file type (Optional - Default RATFOR)
FM - CMS file mode (Optional - Default A1)
```

NEXT keyword

```
NEXT N                        (N=1 by default)
```

Branches to next iteration of Nth loop.

REPEAT-UNTIL statements

```
REPEAT
    RATFOR statement
 UNTIL (legal FORTRAN condition)
```

RETURN

```
RETURN (expression)
```

WHILE statement

```
WHILE (legal FORTRAN condition)
   RATFOR statement
```

*** A RATFOR statement can be any of the following:
1. A single FORTRAN statement.
2. A bracketed set of statements.
3. Any of the RATFOR statements just described.

Quick Reference Guide to RATFOR

(Cosmetics)

* Free form input (ie. spacing is not important).

* Lines ending with = + - * , / | & (are assumed to be continued. No continuation signaler is needed.

* Statements beginining with an all-numeric field is assumed to be a FORTRAN label and is placed in columns 1-5 of the output.

* Strings in matching single or double quotes are converted to Hollerith form ('string'=6Hstring).

* Statement grouping using either { and }, [and], or $(and $).

* Comments beginning anywhere in the input, denoted by #.

* Translation services

==	to	.eq.	~=	to	.ne.
>	to	.gt.	>=	to	.ge.
<	to	.lt.	<=	to	.le.
&	to	.and.	|	to	.or.
~	to	.not.			

RATFOR Reference

Kernighan, Brian W. "RATFOR - A Preprocessor for a Rational FORTRAN", Bell Laboratories Technical Report 55, January 1, 1977.

Using RATFOR on CMS

Running a program through the CMS RATFOR preprocessor is a very simple task. All the user has to do is to type "RATFOR". The system will then ask for the name of the RATFOR file to be processed; the file type must be RATFOR. The user file is then translated to legal FORTRAN, possibly with error messages being flagged by RATFOR. If errors are present, the system will list the error messages on the terminal. Detailed explanations of the RATFOR error messages can be found later in this User's Manual.

If the user program is free of RATFOR errors, it is then compiled using the CMS FORTGI compiler. The compiler flags any illegal FORTRAN statements. The result of the entire RATFOR process is a TEXT file having the same file name as the original user RATFOR program. This TEXT file can be run like any other CMS TEXT file.

While on CMS, if any questions arise as to how to use the RATFOR preprocessor, type "RATFOR ?". The system will respond with a description similar to the one just given. If this does not suffice, follow the sample terminal sessions given later in this document.

Please note that the RATFOR preprocessor may be slow, depending on the length of the user program. Have patience, the preprocessor is not stuck in a loop.

All of the executable files for RATFOR are on the A disk of CMS userid CSDULLES. To gain access to these files, enter the following CMS commands:

```
cp link csdulles 191 333 read all
access 333 g/a
```

Sample Terminal Sessions

'>'   Prompts the user for a response

*** Normal Terminal Session ***

```
>RATFOR
*** RATFOR COMPILER ***

ENTER NAME OF RATFOR FILE:
>TEST


NO RATFOR MESSAGES FOR TEXT

G1 COMPILER ENTERED
SOURCE ANALYZED
PROGRAM NAME = MAIN
*  NO DIAGNOSTICS GENERATED
```

*** Terminal Session with RATFOR errors ***

```
>RATFOR
*** RATFOR COMPILER ***

ENTER NAME OF RATFOR FILE:
>BAD


RATFOR MESSAGES FOR BAD

? error at line 2:
25-Unexpected EOF.
```

The next step is to look up Error Message 25 in the User's Manual for a detailed explanation of the problem.

*** Terminal Session asking for help ***

```
>RATFOR ?
*** RATFOR COMPILER ***

[ Explanation of RATFOR preprocessor ]
```

RATFOR Error Messages
Explanations & Corrections

"Ø1-Missing Left Paren."

Meaning:	A left parenthesis was missing starting an IF, WHILE, or UNTIL statement.
Correction:	Check to ensure that all parentheses are balanced.

"Ø2-Missing Parenthesis in Condition."

Meaning:	The parser encountered unbalanced parentheses in an IF, WHILE or UNTIL conditional statement.
Correction:	Check to ensure that all parentheses are balanced within the conditional.

"Ø3-Illegal BREAK."

Meaning:	An attempt was made to either generate code for the BREAK statement outside of a loop or to transfer control outside of an illegal number of loops (N too high on "BREAK N").
Correction:	Make sure the BREAK statement occurs within a loop. Check the level of the BREAK with the level of looping.

"Ø4-Illegal NEXT."

Meaning:	An attempt was made to either generate code for the NEXT statement while not in a loop, or to transfer control through too many levels of looping (N too high on "NEXT N").
Correction:	Make sure the NEXT statement occurs within a loop. Check the level of the NEXT with the level of looping.

"Ø5-Unexpected EOF."

Meaning:	The parser was expecting more of the input card when an end of file condition occurred.
Correction:	Check for unfinished continuation cards.

RATFOR Error Messages
Explanations & Corrections

"06-Unbalanced parentheses."

Meaning:	The parser encountered unbalanced parentheses in an IF, WHILE, or UNTIL statement.
Correction:	Check to ensure that all parentheses are balanced within the statement.

"07-Missing left paren in FOR statement."

Meaning:	A left parenthesis was expected at the start of the FOR statement.
Correction:	Make sure all necessary parentheses are present.

"08-Unbalanced parentheses in FOR clause."

Meaning:	Unbalanced parentheses were encountered while parsing the FOR statement.
Correction:	Check to see that the FOR statement has a balanced number of parentheses.

"09-FOR clause too long."

Meaning:	The specified FOR clause was longer than the maximum length (currently 200 characters).
Correction:	Where possible, break the FOR clause into smaller pieces.

"10-Non-Alphanumeric name in DEFINE."

Meaning:	The name specified in the DEFINE statement had a non-alphanumeric character in it.
Correction:	Correct the DEFINE to be composed of numbers and letters only.

"11-Definition too long."

Meaning:	The specified definition was longer than the maximum definition size (currently 200 characters).
Correction:	Choose a smaller definition.

RATFOR Error Messages
Explanations & Corrections

"12-Missing comma in DEFINE."

Meaning:	The form of the DEFINE statement having parentheses also has a comma between the name and definition.
Correction:	Add the necessary comma.

"13-Missing right paren in DEFINE."

Meaning:	A definition starting with a left parenthesis did not have a matching right parenthesis.
Correction:	Add the necessary right parenthesis.

"14-GETDEF is confused."

Meaning:	An unexpected token was found in the DEFINE statement.
Correction:	Correct the DEFINE statement according to the rules specified in the language description.

"15-INCLUDEs nested too deeply."

Meaning:	More than 92 user-nested INCLUDE files were opened concurrently.
Correction:	99 is the maximum number of concurrent FILEDEFs allowed by CMS for a FORTRAN file (user-defined files start with unit number 7). Check your program for recursion - since this is not check for, an infinite INCLUDE loop may have occurred.

"16-Specified file does not exist."

Meaning:	The file specified in the INCLUDE statement does not exist.
Correction:	Check to make sure the file name, file type (Default RATFOR), and file mode (Default A1) were spelled correctly.

RATFOR Error Messages
Explanations & Corrections

"17-Error in defining file."

Meaning: The CMS FILEDEF statement did not execute correctly. The specified file was not included.

Correction: Make sure your INCLUDE statement did not include any CMS unprintable characters within the file name.

"18-Token too long."

Meaning: The input token was longer than the maximum token length (currently 200 characters).

Correction: Shorten the token to something within the length boundary.

"19-Missing Quote in string."

Meaning: The string in error was missing its terminating quote mark.

Correction: Add the required quote mark and resubmit the program.

"20-Too many definitions."

Meaning: Too many DEFINE statements were encountered.

Correction: Cut the number of DEFINEs down to the current maximum (6500 characters in all definitions).

"21-Warning: Possible label conflict."

Meaning: A user-defined label may conflict with a RATFOR-generated label.

Correction: Generally RATFOR-generated labels are of the form 23XXX. If a conflict occurs, choose another user label.

RATFOR Error Messages
Explanations & Corrections

"22-Illegal ELSE."

Meaning: An ELSE was encountered that did not have a matching IF statement.

Correction: Check to ensure that every ELSE has an associated IF statement.

"23-Stack overflow in parser."

Meaning: An attempt was made to add too many tokens to the parser stack.

Correction: Send a copy of your RATFOR file to Dick Orgass - CMS userid ORGASS, along with an explanation of the problem encountered.

"24-Illegal right brace."

Meaning: A right brace was encountered that did not have a matching left brace.

Correction: Ensure that all left braces have a matching number of right braces.

"25-Unexpected EOF."

Meaning: The parser was expecting more symbols (possibly loop terminators or brackets) when an end of file condition occurred.

Correction: Make sure that all loops are in accordance with the language description and that every left bracket has a matching right bracket.

"26-Too many characters pushed back."

Meaning: An attempt was made to push more characters on the input buffer than was allowed.

Correction: Send your RATFOR source to Dick Orgass - CMS userid ORGASS, along with an explanation of the problem encountered.

APPENDIX

On Line - Help File

RATFOR

A Rational Alternative to Fortran

RATFOR is a fairly popular preprocessor for Fortran. It provides modern control structures as well as a pleasant syntax for Fortran programs. Many of the truly irritating syntactical conventions of Fortran are avoided in RATFOR and the resulting Fortran code is of good quality. Note that RATFOR will not write any non-standard Fortran unless you write it into the program directly.

Any of the published documentation on RATFOR combined with the material in this file is adequate to use the preprocessor. CMS RATFOR differs from other versions in that lower case letters are mapped into upper case letters except in quoted strings and Hollerith constants. Both single and double quotes may open a quoted string; the next occurrence of the same quote ends the string. Detailed documentaton for this CMS implementation may be obtained by contacting:

Richard J. Orgass
Department of Computer Science
VPI&SU
P. O. Box 17186
Washington, D.C. 20Ø41

CMS userid: ORGASS

Please specify if you want the user's manual or the systems manual. The latter is designed for readers who wish to modify the preprocessor and does not contain information that is needed by general users.

Using RATFOR

The RATFOR preprocessor accepts input files with fixed length records and 8Ø column record length. Tabs are equivalent to blanks so that tabs can be used to provide indented program text that visually extends beyond column 8Ø.

Input files to RATFOR must be of file type RATFOR and located on the user's A disk or a read only extension of this disk. If you have created a file <fn> RATFOR, this file can be converted into a text file for execution by executing the command:

```
ratfor <fn>
```

This will invoke the RATFOR processor and then the Fortran G compiler. If there are no errors, only a file <fn> TEXT will be produced.

If there are RATFOR errors, error messages will be printed on the terminal and written to file <fn> ERROR. The Fortran compiler will not be invoked.

If there are no RATFOR errors but there are Fortran errors, at least two additional files will be produced: <fn> FORTRAN and <fn> LISTING. [There may also be a file <fn> TEXT if one is produced by Fortran.] The listing file will only contain error messages and not the complete text of the program.

Note that the Fortran files produced by RATFOR are very difficult to read; there are no comments and no redundant spaces. These files are only intended for compilation and not for human consumption.

Entering the command:

```
ratfor ?
```

will print this text on your terminal.

Additional Options

Executing the command:

```
ratforh <fn>
```

will invoke the Fortran H compiler instead of the Fortran G compiler to translate the RATFOR program into executable code.

Executing the command:

```
ratno <fn>
```

will invoke the RATFOR preprocessor and will not subsequently invoke any Fortran compiler. If there are no RATFOR detected errors, a single file, <fn> FORTRAN will be created; it contains the Fortran that corresponds to the input RATFOR. If RATFOR detects errors, error messages will be written on the terminal and into file <fn> ERROR.

Both of these commands accept ? as an argument to print this text.

Availability

All of the files needed to use RATFOR are part of the public library of the Computer Science Graduate Program in Northern Virgina. This library is the A disk of userid CSDULLES. To use RATFOR, execute the following commands:

```
cp link csdulles 191 330 read all
access 330 e/a
```

The RATFOR processor assumes that this disk is a read only extension of your A disk; it will not function correctly if this is not the case!

Relevant Files

The following files, located on the A disk of CSDULLES are part of the RATFOR processor:

```
RAT      MODULE
RATFOR   EXEC
RATFORH  EXEC
RATNO    EXEC
QUERYFIL EXEC
RATFOR   HELP
```

The RATFOR source for the preprocessor is available to individuals who might wish to modify the program for their own use. Please contact userid ORGASS if you wish to secure a copy of the source code.